Ex No: 3	**ADVANCED SQL COMMANDS**
Date	08.08.2023

Aim: To execute the given commands making use of aggregate functions, group by clause and order by clause.

Description:

Aggregate Functions:

SQL aggregate functions operate on the multiset of values of a column of a relation, and return a value

The various aggregate functions are:

- **Sum**: - returns the sum of the values.
  Eg: select sum(sal) from emp;
- **Avg**:- returns the average of the values.
  Eg: select avg(sal) from emp;
- **Count**:- returns the number of elements in the collection.
  Eg: select count(*) from emp;
- **Min**:- returns the minimum value in a collection.
  Eg: select min(sal) from emp;
- **Max**:- returns the maximum value in a collection.
  Eg: select max(sal) from emp;

The input to sum and average must be a collection of numbers, but the other operators can operate on collections of non-numeric data types, such as strings as well. The average function will return the average of the given tuple. The aggregation function count is used frequently to count the number of tuples in relation.

Distinct Keyword

To eliminate the duplicates, the keyword *distinct* is used in the aggregation expression. SQL does not allow the use of keyword distinct with count (*) to count the number of records in a relation. It is allowed to use distinct with max and minfunctions, even

though the result does not change.

Eg: Select count (distinct job);

GROUP BY Clause

To apply aggregate function to a group of sets of tuples. The attributes given in the group by clause are used to form groups. Tuples with some value on all attributes in the group by clause are placed in one group.

Eg: Select branch_name,avg(bal) from account group by branch_name;

ORDER BY Clause

This clause causes the tuples in the result of a query to appear in sorted order. We specify asc for ascending order and desc for descending order.

Eg: Select * from loan order by amount desc;

Advanced SQL Queries:

1. Find the average price of confirmed tickets

   Query: select avg(Price) from Ticket_1128 where status = 'Confirm';

```
SQL> SELECT AVG(price)
  2  FROM Ticket_1128
  3  WHERE status = 'confirmed';
     47.75
```

2. Display the starting date as "Start" of all the events form events table.

   Query: SELECT event_id, TO_CHAR(start_date, 'DD-MON-YYYY HH:MI:SS') AS "Start" FROM Events_1128;

```
SQL> SELECT event_id, TO_CHAR(start_date, 'DD-MON-YYYY HH:MI:SS') AS "Start"
  2  FROM Events_1128;
       101 10-SEP-2023 06:00:00
       102 15-SEP-2023 09:30:00
```

3. Find the Minimum date, maximum date of all the events, and the number of months in between the min and max dates
   Query: SELECT MIN(start_date) AS "Minimum Date",
   MAX(start_date) AS "Maximum Date",
   MONTHS_BETWEEN(MAX(start_date), MIN(start_date)) AS "Months Between"
   FROM Events_1128;

```
SQL> SELECT MIN(start_date) AS "Minimum Date",
  2         MAX(start_date) AS "Maximum Date",
  3         MONTHS_BETWEEN(MAX(start_date), MIN(start_date)) AS "Months Between"
  4  FROM Events_1128;
10-SEP-23 15-SEP-23     .149865591
```

4. Find the total number of venues used to conduct the events
   Query: SELECT COUNT(DISTINCT venue_id) AS "Total Venues" FROM Events_1128;

```
SQL> SELECT COUNT(DISTINCT venue_id) AS "Total Venues"
  2  FROM Events_1128;
           2
```

5. Find the number of users in User table
   Query: SELECT COUNT(*) AS "Number of Users FROM User_1128;

```
SQL> SELECT COUNT(*) AS "Number of Users"
  2  FROM User_1128;
              2
```

6. Find the length of password of User = p@ssw0rd from User table
   Query: SELECT LENGTH(password) AS "Password Length" FROM User_1128 WHERE password = 'p@ssw0rd';

```
SQL> SELECT LENGTH(password) AS "Password Length"
  2  FROM User_1128
  3  WHERE password = 'p@ssw0rd';

Password Length
---------------
              8
```

7. Concatenate the data and time of Event table as “Date-Time” and display Date-Time and the name of the event if the 5th character of event name is ‘e’
   Query: SELECT event_name, TO_CHAR(start_date, 'DD-MON-YYYY HH:MI:SS') || ' ' || event_name AS "Date-Time" FROM Events_1128 WHERE SUBSTR(event_name, 5, 1) = 'e';

```
SQL> SELECT event_name,
  2         TO_CHAR(start_date, 'DD-MON-YYYY HH:MI:SS') || ' ' || event_name AS "Date-T
ime"
  3  FROM Events_1128
  4  WHERE SUBSTR(event_name, 5, 1) = 'e';

EVENT_NAME
--------------------------------------------------------------------------------
Date-Time
--------------------------------------------------------------------------------
Concert A
10-SEP-2023 06:00:00 Concert A

Conference B
15-SEP-2023 09:30:00 Conference B
```

8. Find the user names whose name ends with “pez” from user table
   Query: SELECT username FROM User_1128 WHERE SUBSTR(username, -3) = 'pez';

```
SQL> SELECT username
  2  FROM User_1128
  3  WHERE SUBSTR(username, -3) = 'pez';

no rows selected
```

9. Left pad the seat number of Ticket table with “000”
   Query: UPDATE Ticket_1128 SET seat_number = LPAD(seat_number, 3, '0');

```
SQL> UPDATE Ticket_1128
  2  SET seat_number = LPAD(seat_number, 3, '0');

2 rows updated.
```

10. Display the event details conducted at the same Venue_ID
    Query: SELECT * FROM Events_1128 e1 WHERE EXISTS (SELECT 1 FROM Events_1128 e2 WHERE e1.venue_id = e2.venue_id AND e1.event_id <> e2.event_id);

```
SQL> SELECT *
  2  FROM Events_1128 e1
  3  WHERE EXISTS (
  4      SELECT 1
  5      FROM Events_1128 e2
  6      WHERE e1.venue_id = e2.venue_id
  7      AND e1.event_id <> e2.event_id
  8  );

no rows selected
```

11. Find out the number of venues in each country from Venue table
    Query: SELECT country, COUNT(*) AS "Number of Venues" FROM Venue_1128 GROUP BY country;

```
SQL> SELECT country, COUNT(*) AS "Number of Venues"
  2  FROM Venue_1128
  3  GROUP BY country;

COUNTRY                                            Number of Venues
-------------------------------------------------- ----------------
USA                                                               1
Canada                                                            1
```

12. Add a column named “Remarks” in Venue table. Fill the remarks column with “No Remarks” values using NVL2 command. Print Venue_ID, Country, and Remarks column.
    Query: ALTER TABLE Venue_1128
    ADD remarks VARCHAR2(100);

    UPDATE Venue_1128
    SET remarks = NVL2(remarks, remarks, 'No Remarks');

```
SQL> UPDATE Venue_1128
  2  SET remarks = NVL2(remarks, remarks, 'No Remarks');

2 rows updated.
```

13. Use round and trunc functions to round off and truncate the value 25.235789 to 2 decimal positions using dual table.
Query: SELECT ROUND(25.235789, 2) AS "Rounded Value", TRUNC(25.235789, 2) AS "Truncated Value" FROM DUAL;

```
SQL> SELECT ROUND(25.235789, 2) AS "Rounded Value",
  2         TRUNC(25.235789, 2) AS "Truncated Value"
  3  FROM DUAL;

Rounded Value Truncated Value
------------- ---------------
        25.24           25.23
```

14. Prefix price column with a value of 0 in ticket table to make the length of price =10 digits.
Query: UPDATE Ticket_1128 SET price = LPAD(price, 10, '0');

```
SQL> UPDATE Ticket_1128
  2  SET price = LPAD(price, 10, '0');

2 rows updated.
```

15. Retrieve all events ordered by date and time in ascending order.
Query: SELECT * FROM Events_1128 ORDER BY start_date;

```
SQL> SELECT *
  2  FROM Events_1128
  3  ORDER BY start_date;

  EVENT_ID
----------
EVENT_NAME
--------------------------------------------------------------------------------
START_DAT   VENUE_ID
--------- ----------
       101
Concert A
10-SEP-23        301

       102
Conference B
15-SEP-23        302

  EVENT_ID
----------
EVENT_NAME
--------------------------------------------------------------------------------
START_DAT   VENUE_ID
--------- ----------
```

Result:

The basic sql commands are executed successfully and displayed using sql.